**FB9406V2**

# ECS Project Object-Oriented Technology Transition Plan

**White Paper**
***Working Paper***

**April 1994**

Prepared Under Contract NAS5-60000

**RESPONSIBLE ENGINEER**

| Audrey B. Winston /s/ | 4/25/94 |
|---|---|
| Audrey Winston, Quality Office<br>EOSDIS Core System Project | Date |

**SUBMITTED BY**

| Michael S. Deutsch /s/ | 4/26/94 |
|---|---|
| Michael S. Deutsch, Quality Office Manager<br>EOSDIS Core System Project | Date |

Hughes Applied Information Systems, Inc.
Landover, Maryland

This page intentionally left blank.

# Contents

## 1. Introduction



## 2. Background



## 3. Object-Oriented Implementation Strategies

## 4. Segment Object Technology Plans



## Figures



## Tables



## Abbreviations and Acronyms

# 1. Introduction

## 1.1 Purpose

This document presents a low risk strategy and plan for transitioning object-oriented concepts and technology into the Earth Observing System Data and Information System (EOSDIS) Core System (ECS). This plan is submitted in response to a NASA request to *"...provide an object-oriented design transition plan that addresses risk reduction and is coordinated with baseline and future technologies."* It addresses how the object-oriented paradigm in general, and the Object Modeling Technique (OMT) specifically, will be implemented on the program. Risk is minimized by introducing object-oriented concepts and techniques in a start small and grow incrementally approach.

The transition plan includes both near term and long term strategies. In the near term an **Early Adopters** strategy will be instituted within each Segment. Segment early adopter plans and lessons learned will be coordinated by an **Object-Oriented Core Team**. The core team includes representatives from each of the segments, System Integration and Planning (SI&P), and is lead initially by the Software Engineering Process Group (SEPG). Responsibility for leading the core team may shift in time as the focus of the core team migrates from software early adopters to system wide object technology integration. This structure facilitates coordination and visibility into the controlled adoption of object technology.

Another facet of the near term strategy blends classroom instruction, on-the job-training, and consulting into an innovative concept know as the **Fast-Track.** This facilitate rapid learning and assimilation of the technology into a development team. First, candidate projects are screened by an object-oriented risk assessment checklist. Second, just-in-time training in object-oriented analysis and design is provided by industry experts. Finally, on-the-job consulting and mentoring is provided to ensure successful take up of the technology by the team.

The long term strategy is to facilitate the adoption of object technology at the system design level where the maximum benefits can be realized. We will develop a single methodology which incorporates essential aspects of the Process for System Development methodology along with features of the OMT methodology. A single hybrid methodology will eventually be reflected in the System Engineering Management Plan, System Implementation Plan, and other program documents.

The long term plan also calls for a well defined software process with automated CASE support. The Software Development Plan will be updated to define the object-oriented methodology details. Updates will address process steps, metrics, design review criteria, document annotated outlines for deliverable documentation, and quality assurance criteria.

## 1.2 Organization

This paper is organized as follows:

- Section 1 presents the purpose of the document, its organization, and logistics concerning its review and points of contact.
- Section 2 explains why the ECS project should introduce the object-oriented paradigm.
- Section 3 presents the program's low risk strategy to infuse object-oriented concepts and techniques.
- Section 4 extracts relevant information from each Segment's development plan regarding the introduction of object-oriented technology to facilitated risk management and lessons learned sharing.

## 1.3 Review and Approval

This White Paper is an informal document approved at the Office Manager level. It does not require formal Government review or approval; however, it is submitted with the intent that review and comments may be forthcoming.

This is the second delivery of the object-oriented technology transition plan white paper. It provides the necessary background and strategy to understand the transition framework and overall strategy. It further develops these concepts into detailed tasks, milestones, and checkpoints. It should be noted that the introduction of object-oriented technology is already underway in the form of technology prototypes. This revision provides some specific recommendations on the software engineering development process including modifications to review criteria, metrics for object-oriented design, schedule guideline, and task definitions. Therefor, the segment specific plans presented in this document represent work in progress and are subject to change.

Questions regarding technical information contained within this paper should be addressed to the following ECS and/or GSFC contacts:

- ECS Contacts

  Audrey Winston, Quality Office
  Software Engineering Process Group
  (301) 925-0353

  Allan Bowers, SI&P Office
  Software Engineering Process Group
  (301) 925-0643

- GSFC Contact

  Gail McConaughy
  System Engineering Office Manager
  (301) 286-7741

Questions concerning distribution or control of this document should be addressed to:

Data Management Office
The ECS Project Office
Hughes Applied Information Systems, Inc.
1616A McCormick Dr.
Landover, MD 20785

# 2. Background

## 2.1 Selecting Object-Oriented Technology

In response to science community and NASA direction, and as a result of post-SRR discussions and analyses, the ECS project has accepted what could be interpreted as a "mandate" from the scientific community to develop a new conceptual architecture capable of meeting the needs of the science community today and in the future. Together, the elements of the mandate comprise the building blocks for an evolutionary system. That mandate is detailed in the EOSDIS Core System Science Data Processing Subsystem Reference Architecture (FB9401V2). Object technology is well suited to the needs of the new reference architecture.

The original ECS development strategy was based on an incremental waterfall process with formal reviews and documentation being delivered during each increment. This original plan also included the concept of evaluation packages to enhance early user feedback on requirements and design alternatives. Evaluation packages could contain both prototype and incrementally developed software. This means that a product could be in various states of formality at different points in time. The ideal solution would be to provide a methodology which supports an iterative development cycle including migration from prototype to formal development at any time. It would also facilitate recursive passes through the analysis, design and implementation activities with minimal breakage. The structured analysis and design methodology will permit this round trip process but requires more rework on each pass than does the object-oriented approach. The time and engineering effort required to maintain all of the required documents in the face of a migrating baseline would be, at best, expensive and difficult to manage.

Recent evidence has confirmed that failure of many large and complex software development projects can be attributed to the shortcomings in the process implied and encouraged by structured methodologies. These failures include:

- an inability to validate the user's requirements resulting in considerable changes later
- high maintenance costs due to latent system wide problems being discovered in integration and post deployment when the cost to repair is high
- duplication of effort from phase to phase and from project to project. The top down approach imposed by structured methods creates barriers to reuse of existing software products.

The ECS project has already initiated several risk mitigation strategies to deal with these issues including the recent Multi-Track Development Plan. The Multi-Track Development Plan (White Paper WP9404V2). The Multi-Track Development Plan for the ECS Project includes several development process models including prototype, incremental, tool, evaluation package, and formal releases. Object-oriented technology, with its direct support for iterative development, complements rather than complicates the Multi-Track Development Plan.

In addition to the shortcomings implied by the process model, structured techniques have inherent software engineering weaknesses. These weaknesses include:

- **Multiple transformations in design evolution.** A dependence on a transformation approach to discovery instead of an elaboration approach results in higher developments and maintenance costs. Every time a requirements model (Data Flow Diagram) is transformed into a different design notation (Structure Chart), and a design notation is transformed into code or other implementation notations, engineering effort is required to restate the same information in different syntax and semantics. This translation effort does not add value to the product; it results in non-productive or wasted effort.
- **Separated data modeling.** When data modeling is separated from behavior modeling, the process results in higher integration and maintenance costs as well as reduced adaptability to change over time. Each time data definitions are changed all processes utilizing the data must be evaluated for impact.

The object-oriented paradigm, together with the multi-track process models, overcome most of the short comings of both the monolithic development process model and the structured analysis and design techniques. The object-oriented paradigm is based on a number of fundamentally different principals which facilitate the benefits of the technique. They include:

- **Encapsulation**. Data and their associated processing are hidden to outside users of the data. This aids greatly in localizing potential problems and limits the scope of change impact.
- **Inheritance and polymorphism**. Together these techniques permit the factoring out of common information structures and behavior so that it can be defined once, encapsulated, and reused within the architecture.
- **Abstraction**. Systems may be modeled, evaluated, and executed at various layers of abstraction. This greatly facilitates localizing the impact of change and facilitates reuse.

Together the object-oriented principals facilitate a more iterative development process. The resulting product is deliverable earlier, more extensible, less costly to maintain, and amenable to reuse.

## 2.2 Rationale for Selecting OMT

The benefits of the object-oriented paradigm could be limited to the implementation phase by introducing an object-oriented language or could be leveraged into the analysis and design activities by introducing an methodology based on object-oriented principles. C++, a hybrid language between procedural and object-oriented programming, is the projects standard language. This permits object-oriented constructs to be implemented while not mandating their application in the implementation. To gain the advantages of object technology in the earlier development phases an object-oriented methodology is needed. Of the half dozen or so prominent methodologies Object Modeling Technique (OMT) was selected as the base methodology for the ECS project. OMT is well documented, supported by several CASE tool and consulting vendors, and certified by the Object Management Group (OMG). OMT is a hybrid methodology. It directly supports all of the object-oriented constructs (encapsulation,

inheritance, polymorphism, aggregation etc.) while permitting considerable flexibility in the development process. It is one of the most robust methodologies in the market place. To understand how the methodology accomplishes its flexibility it is mapped to a set of continua. Each continuum presents divergent points of view on aspects important to methodology comparisons. It will be shown that OMT supports the most appropriate end, or the widest range, of alternatives across each evaluation continuum.

### 2.2.1 Discovery Approach

The technique used to "discover" a design or implementation from a problem statement requires a designer to select an approach to migrate through various levels of abstractions. The two ends of the discovery approach continuum are elaboration and transformation, figure 1. Elaboration is typified by successive refinements of the same information using the same language semantics until an implementation is discovered. Transformation refers to the restatement of one domain language such as data flow modeling into another domain language such as Structured Query Language (SQL).

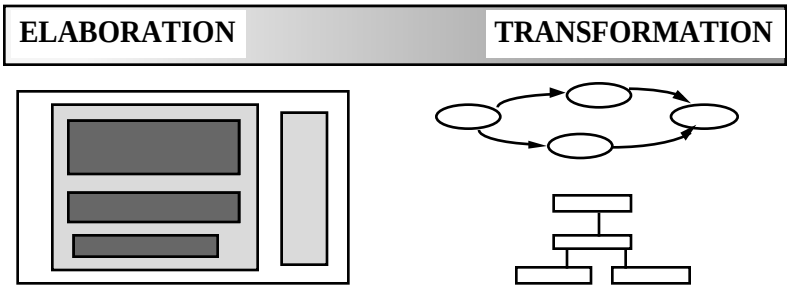



***Figure 1. Object Discovery Continuum***

Traditional software development is based on the transformation of requirement model semantics [such as processes and data flows] into design model semantics [such as control structures and data schema]. This technique is based on the assumption that design architectures are fundamentally different from their respective analysis model. That is, different structures and languages are needed to express the designers understanding of the problem. It also requires that the analysis be essentially completed prior to design architecture commitment. The architecture can not be fully described until all of the raw material is available from the requirements analysis activity. Changes to the requirements usually mean a significant effort to change the design architecture. In structured techniques each time the requirements are transformed from one language and set of structures to another engineering time and energy are lost. There is no value added to the resulting design simply for having done the transformation. This is not the case in the OMT methodology.

The elaboration approach to discovering objects is an evolutionary process. Real-world entities start out as objects in the requirements model and are maintained as objects is the design model. The analysis and design model are essentially the same model with different levels of detail.

Additional attributes, operations and implementation objects are added as the design is elaborated but the original real-world objects are retained throughout the development cycle. No information is ever lost or transformed into a different set of semantics. This preserves the real world view of the problem and avoids the unnecessary transformation to different design semantics.

### 2.2.2 Single vs. Multiple Semantics Continuum

Semantics refers to the language, graphics or words, used to communicate within a given domain; analysis, design, or implementation, Figure 2. Single semantics simply mean the engineer can use a single set of notations and rules from analysis through design. Multiple semantics refers to the use of specialized languages for specialized problems. Multiple semantics are needed in non-object-oriented domains. Relational databases with their data definition language (DDL) and data manipulation language (DML) do not map directly to real world concepts but are necessary where legacy systems are employed. Wherever multiple semantics are employed added effort is needed to bridge the "semantic gaps."

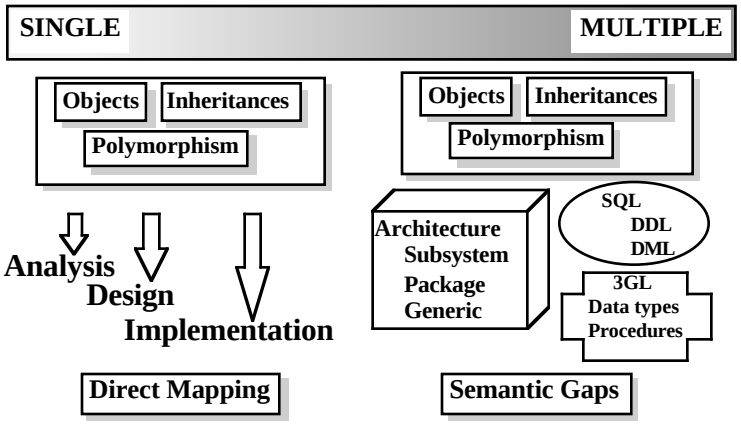



***Figure 2. Single vs. Multiple Semantics Continuum***

A superior approach to multiple notations would be to permit a single notation to be used for both requirements modeling and architecture design modeling. The semantics of objects, encapsulation, inheritance, cardinality, associations, aggregation, and polymorphism should be directly supported in a single notation. OMT, unlike structured methodologies and some other object-oriented methodologies, utilizes a single, easy to use notation set for both analysis and design modeling. The notation provides direct support for all of the object-oriented concepts.

The single notation of OMT allows a layered virtual machine approach to both the requirements definition and the design. This provides continuity, precise and concise, from problem statement to implementation--nothing is ever lost. The OMT notation can be directly translated into C++ language specifications [or headers]. Since the notation does not change from analysis to design the developer does not incur additional costs for recursing backward into analysis from design or from implementation to design. It facilitates prototyping and development phase iteration.

### 2.2.3 Object Realization Continuum

Object realization refers to the process used to discover or identify objects in the application domain, figure 3. Pure object-oriented methodologies such as Object Behavior Analysis or Class Responsibility Collaboration focus mainly on the behavior of the application. Structured methodologies and some object-oriented methodologies such as Shlaer-Mellor or Coad-Yourdon focus primarily on data structures and data transformations. Neither of these extremes will satisfy the wide range of ECS problem domains.

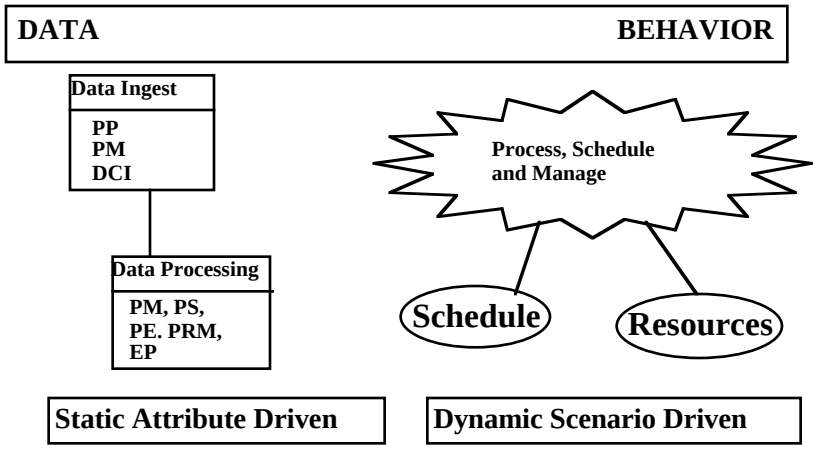



***Figure 3. Object Realization Continuum***

The data structure oriented identification approaches focus on internal structures first and external communications second. This is characteristic of information models such as entity-relationship models. Operations are added to perform data transformations and movement of data between objects. The major advantage of this technique is that it is a small step into the object-oriented domain. The major disadvantage is the increased coupling between objects and the lower resulting cohesion within an object.

The behavior oriented identification approach, on the other hand, focuses on objects needed to carry out a scenario driven responsibility or service. This approach allocates system behavior to objects. Since attributes are added later in the process there is less opportunity for attribute coupling between objects. It also tends to produce fewer classes and hence, less source code to develop and maintain. It is generally accepted in the industry that people require a longer time to make the paradigm shift to a pure object-oriented mind-set.

The pure object-oriented or behavior oriented approach is known to produce smaller and less complex software systems. In a study conducted by Boeing two object-oriented methodologies were compared. One used a data driven process while the other used a behavior driven process. Metrics appropriate to object-oriented designs were collected and analyzed. The measures in the study included:

- Lack of Cohesion among object Methods (LCOM)
- Weighted Attributes per Class (WAC)
- Weighted Methods per Class (WMC)
- Depth of Inheritance Tree (DIT)
- Number of Children (NOC)
- Coupling Between Objects (CBO)
- Violations of Demeter (VOD)
- Response for a Class (RFC)

The study results figure 4, clearly suggest that a more behavior-oriented process approach will produce a less complex system, at least in terms of object-oriented complexity measures.

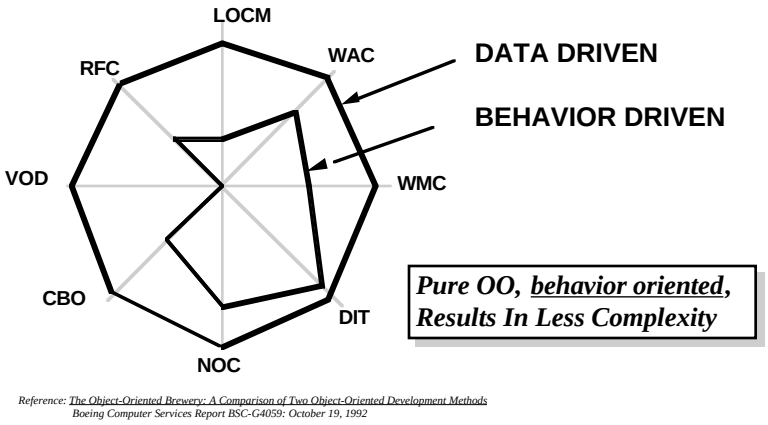



***Figure 4. Methodology Complexity Comparison***

The ECS program contains a range of object realization or identification needs. The OMT methodology process is well defined yet sufficiently flexible to facilitate a focus on either data or behavior. The notation includes support for identification of both public and private attribute structures and behavior specifications. The choice of which aspect to focus on is left to the engineers and will be driven by both the degree of object-oriented understanding in the team and the nature of the problem.

### 2.2.4 System Vs Local Problem Discovery Continuum

When, in the development cycle, problems are discovered greatly influences the maintenance costs of the resulting system. The greater the scope of impact the discovered problem has the greater the maintenance costs. The ideal methodology would encourage developers to find all problems, but in particular system wide problems early, in the analysis and design phases.

Systems developed using a structured methods have a peculiar recurring pattern in terms of finding errors. The distribution of system level problems Vs module level problems was studied by CMU on a project titled "Interactive Pittsburgh", figure 5. Engineers tend to understand the part of the system they are working on and resolve local or module level problems quickly. Most module level problems are removed by the time the system moves into integration. System level problems tend to follow exactly the opposite rate of discovery. Problems which require knowledge of many parts of the system are usually discovered later in the life cycle. This traditionally results in more effort being applied to the system teat and integration phases.

The OMT approach produces a very different profile of problem discovery and system stability. Objects are encapsulated. This limits the scope of any problems they may contain. The analysis and design models are essentially two different views of the same model. The design is a refinement of the analysis model but both share the same information; this facilitates early identification of problems.

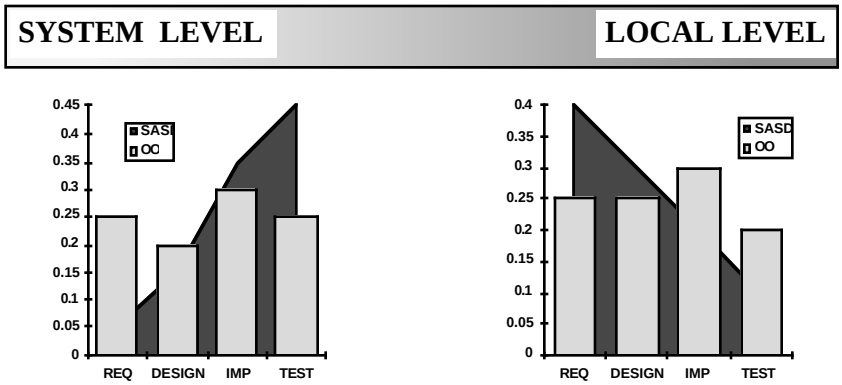



***Figure 5. Problem Discovery Continuum***

### 2.2.5 Case Tool Support

The OMT methodology is supported by a number of vendor CASE tools. For the ECS program the selection criteria included not only support for the methodology but more importantly would the tool provide features which would allow it to scale up to the size and complexity of the ECS program? The Software through Pictures product by IDE was selected based on a paper analysis. Then, a hands-on evaluation was conducted over a period of 30 days by representatives from several different subsystems. The tool supports configuration management of models, multiple views of the same model, automated collection and reporting of some object-oriented metrics, multi-user access, and other features critical to the successful application of the methodology to the ECS program.

## 2.3 Related Documents

This document focuses on the rationale for selecting OMT as the basis for the program object-oriented methodology and the plan to transition parts of the program from structured design to object-oriented design. It does not repeat information provided in other program documents. Justification is derived from concepts developed in related project documents, including:

| | |
|---|---|
| 193-00623 | ECS Evolutionary Development White Paper, December 1993 |
| 194-813-SI4-002 | Planning and Scheduling Prototype Results Report for the ECS Project, February 1994 |
| FB9401V2 | EOSDIS Core System Science Data Processing Subsystem Reference Architecture White Paper, March 1994 |
| FB9403V1 | Release Plan Content Description White Paper, February 1994 |
| FB9404V2 | Multi-Track Development for the ECS Project White Paper, March 1994 |

# 3. Object-Oriented Implementation Strategies

## 3.1 Overview

To maximize the benefit of the technology without jeopardizing the program, OMT will be introduced in a managed risk approach. Two parallel paths are being pursued, a near term strategy for early adopters of the technology and a long term strategy to facilitate eventual implementation into all appropriate parts of the program. These strategies are captured in this plan and then coordinated and maintained by an ***Object-Oriented Core Team.***

The transition strategy covers:

***Table 1. Transition Strategy***

| Strategy Topic | Near Term | Long Term |
|---|---|---|
| Object-Oriented Core Team | | O |
| Object Technology Training | O | |
| Consulting & Mentoring | O | O |
| Fast-Track | O | |
| System Engineering Methodology | | O |
| Software Engineering Methodology | | O |

## 3.2 Object-Oriented Core Team

To effectively coordinate and integrate implementation strategies across the project, dedicated resources, with the appropriate disciplines, will form an ***Object-Oriented Core Team***. The team will consist of SEPG members from the following organizations:

***Table 2. Object-Oriented Core Team***

| Organization | Role/Responsibility |
|---|---|
| Quality | Core Team Leader |
| Quality | Software Development Plan |
| System Integration & Planning | System Engineering Plan |
| System Integration & Planning | System Implementation Plan |
| Flight Operations Segment | Represent FOS Implementation |
| Science Data Processing Segment | Represent SDPS Implementation |
| Computer System Management Plan | Represent CSMS Implementation |

The ***Object-Oriented Core Team*** will have responsibility for the following activities:

- coordinate knowledge sharing to further evolve transition plans and infrastructure support
- assess opportunities to implement the technology including benefits and risks identification
- distribute examples including documentation, CASE tool models, and process descriptions developed by Early Adopters
- review and recommend changes to the System Engineering Management Plan, System Implementation Plan, Software Development Plan and other affected program documents.

## 3.3 Object Technology Training Program

OMT represents a new way of thinking for most engineers, scientists, and managers involved in the ECS program. Hence, more than simply training courses, our transition plan calls for a training process. Classroom training for engineers will be combined with on-the-job-training by mentors. Managers and customers will also receive specialized training. Over time, as people become skilled in the technology, more specialized and advanced training will be provided. We foresee the need for specialized training in object-oriented project management, test, metrics, design standards and reuse strategies and libraries.

All engineers will receive their training regarding OMT, Software through Pictures, and C++ on a just-in-time basis. To get the most economic benefit from the training, it is important that individuals are trained just in time to apply the material to avoid knowledge loss due to latency effects. The training courses selected for object-oriented analysis and design contain both theory and skill building. To the extent possible, the training will be tailored to work on program specific examples for both analysis and design.

A training program has been established which addresses, in addition to engineering, all levels of management and the customer. Training courses include:

- Executive Management Overview (half-day)
- Overview of Object-Oriented Methodology (2 days)
- Object-Oriented System Engineering (3 days)
- Object-Oriented Analysis (4 days)
- Object-Oriented Design (4 days)
- C++ Programming (4 days)
- Other specialized courses such as OO Project Management, Designing for Reuse etc. will be added as the requirements and their cost effectiveness are identified. These may be in-house developed course or contracted courses.

The training plan details are contained in the ECS Development Facility Training Plan Project Instruction, SD-1-002. It contains the details of who should be trained in what topics and when

training is scheduled to take place. This training plan will evolve to provide more specialized courses as identified by the needs analysis routinely conducted by the SEPG.

## 3.4 Consulting & Mentoring Program

Studies have indicated that the use of a mentor-apprentice scheme to adopting object-oriented technology will result in better long term effects. While it may appear to be more cost effective to sequester the trained engineers in a core team, the anticipated pervasive nature of the technology dictates a different tack. In the short term we will take advantage of industry expert consultants. In addition to methodology mentoring the consultants will help in assessing when and how to introduce object-oriented concepts into the project. In the long term we expect each segment to develop their own mentors.

Fast-Track teams are encouraged to include a full time consultant as part of the team for their early iterations. Based on industry experience, this approach provides the following advantages:

- recovers much of the schedule time normally lost to the first application of object technology
- improves the quality and "object-orientedness" of the Fast-Track product
- reduces the learning time for the Fast-Track team members
- lowers risk associated with adopting a new technology.

Once engineers have been through all the requisite training and have successfully built an object-oriented application, they will be able to act as mentors on subsequent projects. This will result in a geometric progression of trained engineers over time.

## 3.5 Fast-Track Strategy

The Fast-Track strategy goal is to cost effectively leverage object-oriented technology training and skills into carefully selected parts of the program with relatively low risk. Fast-Track pilot/training projects will cover the entire life cycle of activities in a short period of time, nominally three months. The strategy is to combine just-in-time training in objected-oriented analysis and design with on-the-job training and consulting, while developing real project products. This has been shown to be the most effective way to build expertise.

### 3.5.1 Fast-Track Activities

The specific activities of the fast track strategy are:

1. Select a candidate application and perform a risk assessment. Guidelines for selecting and assessing a candidate application are included in section 3.5.2.
2. Scope the project to be approximately three months so that all life cycle phases can be visited during the Fast-Track.

3. Combine just-in-time classroom training with mentoring and on-the-job-training on a small scale project in the following sequence:
    A. four days of object-oriented analysis classroom training
    B. two to eight weeks of on-the-job analysis on the selected application with mentoring
    C. four days of object-oriented design classroom training
    D. complete the design, implementation, and testing with consulting/mentoring support.
4. Capture lessons learned and recommendations for future iterations and disseminate through the core team.

### 3.5.2 Fast-Track Risk Assessment

The best return on investment will come from projects where object technology can be introduced early, have a long life span, requirements are expected to evolve over time, are relatively complex, and desire to take advantage of reuse. Longer life spans can better afford the start-up investment costs normally associated with learning, and adapting to, a new technology. Object modeling technology aids management in controlling complexity by facilitating better communications across development teams by enforcing encapsulation and abstractions. It also facilitates better customer communications by modeling real world objects rather than computer science or implementation abstractions.

Selecting a Fast-Track opportunity includes considerations for:

- Determining when to begin introducing the technology in the development cycle
- Characterizing the risk factors of the opportunity
- Establishing a plan with goals, objectives, and demonstrable milestones

#### 3.5.2.1 When to Begin

The earlier in the life cycle object-oriented technology is introduced, the easier it is to transition back to structured techniques and the higher the potential return on investment. It is relatively easy to transition from an object-oriented analysis and design model to a non-object-oriented implementation should the risk indicators suggest this is necessary. It is more difficult to transition from a structured approach to an object-oriented approach at any phase in the life cycle. The best risk managed approach is to start as early as possible in the requirements modeling activity with an object-oriented model and only transition back at the implementation phase if performance or other implementation constraints dictate the need to relax object-oriented constructs.

#### 3.5.2.2 Characterizing An Opportunity

Object oriented technology is applicable in all ECS problem domains. However its greatest utility can be expected from projects with the following characteristics:

- highly interactive, user-centric--not batched
- complex navigation of complex information structures--not massive retrieval of simple structures
- event drive architectures--not sequential flow of control
- client/server and distributed applications--not monolithic
- workstation or larger based applications--not small embedded processors
- small team size developments in early phases (4-9 people)--not large scale until the organization (engineering and management) have gained experience in the technology

Object technology is least useful in the design of mathematical algorithms and hard real-time interrupt managed applications. However, object technology may still be useful in the analysis or implementation (hybrid C++) phases of these applications.

Candidate projects for the Fast-Track will represent a diversity of project problem domains. This approach affords the best opportunity to learn about all aspects of applying the technology. Candidates will be selected from each of the segments: FOS, SDPS, and CSMS. Problem domains which will be explored include:

- real-time and high performance processing
- instrument planning and scheduling
- information retrieval and storage management
- communications infrastructure mechanisms including distributed processing
- graphical user interface

For each candidate target of opportunity to apply object-oriented technology the following risk factors should be rated as High, Medium, or Low risk. For High and Medium risk items a mitigation strategy should be included in the project's plan.

- **Skills of the available people**

    1. Do they understand the concepts of modern software engineering such as encapsulation, information hiding?
    2. Do they have object-oriented or object-based language skills such as Small talk, C++, or Ada?
    3. Do they all have domain knowledge?

- **Object technology support**

    4. Is a mentor part of the team or readily available?
    5. Is training available for those not already trained?

- **Scope of the project**

  6. Are the requirements well understood by the team?
  7. Is the problem well bounded?

- **Development schedule**

  8. Is the size of the project scoped to less than three months for the first iteration?
  9. Is this pilot on the program's critical path?

- **Management buy-in**

  10. Is management at all levels aware of the plan?
  11. Is management involved in the cultural change?
  12. Is management committed to following through with the support required for success?

#### 3.5.2.3 Establish a Fast-Track Plan

Once a target project is selected, a risk assessment has been completed, the team members are identified, and the training schedule is aligned, the project plan needs to be completed. This includes the identification of activities, milestones and risk assessment check points. A schedule template and checklist of activities which may be used as a starting point is contained in the Advance Concepts Center Overview of Object-Oriented Methodology. The specific section to examine are:

- Section 110-720 Analysis Phase--Milestones and Scheduling
- Section 110-730 Design Phase--Milestones and Scheduling

Next, select a stable requirements baseline for the training pilot and scope the project to be completed in approximately three months. Include demonstrable milestones related to the critical success factors for the pilot. Have the plan reviewed by the Object-Oriented Core Team for risk assessment and coordination assistance of consulting and other support identified in the plan. Ensure a consultant or mentor is available and part of the plan.

At each major milestone, record lessons learned and recommendations for successive project increments and for dissemination by the core team to other parts of the organization.

## 3.6 System Engineering Methodology

The development of a System Engineering Methodology using object-oriented concepts must consider that the object-oriented concepts are not as established in System Engineering as they are in Software Engineering. The ECS program is using the object-oriented concepts in the development of the System Design. However, the fully defined and agreed to object-oriented System Engineering methodology is still under development. The approach to developing an object-oriented System Engineering process for ECS is discussed below with a near-term perspective, i.e. before SDR, and long-term perspective, after SDR.

***Table 3. Activities for Developing Object-Oriented System Engineering Methodology***

| Near-Term (before SDR) | Long-Term (after SDR) |
|---|---|
| System Design using object-oriented concepts | Document "as-built" System Design process |
| Update of System Engineering Plan | Segment Design using object-oriented concepts |
| | Update System Design Specification |
| | object-oriented SE Training Material |

In the near-term, two activities are critical: 1) development of a System Design using object-oriented concepts, and 2) updating the System Engineering Plan. As described in ECS SDPS Reference Architecture White Paper, the System Design is being developed, at least partly, using object-oriented concepts. The determination of a consistent set of object-oriented concepts used to describe the entire system will be resolved as part of the development of the System Design Specification. The System Engineering Plan is to be updated prior to SDR based upon comments of the first version which was submitted to EOSDIS, and based upon two recent white papers: The ECS Evolutionary Development Process and Multi-Track Development for the ECS Project. In addition, the revision of the plan can begin to incorporate object-oriented ideas based upon the Segment Design Document (SDS) development plan and the recent class on object-oriented System Engineering.

The long-term strategy is based upon maturing the definition of an ECS System Engineering approach based on object-oriented concepts in support of the evolutionary nature of the ECS program. A baseline of this process will be set by the activities leading up to SDR. It will be critical to capture this process along with lessons learned to refine the ECS System Engineering process. This may follow the model of a white paper - ECS object-oriented System Engineering Process - followed by updates to the System Engineering Plan. Although there is only one SDR on the ECS program, because of the multiple releases in the program and the need to conduct object-oriented SE for some segments leading up to PDR, a revision to the System Engineering Plan is warranted. Besides the as-built System Design process, new information from ACC, information available from the National Council on Systems Engineering and other sources will be used to update the System Engineering Plan.

## 3.7 Software Methodology

The impact of introducing object oriented technology is pervasive. People, products, and processes are all affected. All three aspects must be considered together. Integrating the technology into products without considering the people, including management and customer, would not maximize the benefits afforded by the technology. Management must be able to monitor and control project development process performance as well as product quality. Therefore, amendments to the software development plan and other project guidelines will address each of these areas in a balanced way.

- Transitioning People to Object Technology

- Transitioning Products to Object Technology
- Transitioning Processes to Object Technology

### 3.7.1 Transitioning People to Object Technology

To assist the cultural change to object technology two aspects must be addressed; organizational roles and communications. Domain expert, mentor, and tool smith are examples of object-oriented team roles recognized in the industry as having different responsibilities in dealing with object technology. These recommendations will be examined and integrated into the program's organizational structure where appropriate.

A common understanding of terms and concepts is very important in adopting any new technology. A series of management and customer briefings will be conducted to ensure we are all speaking the same language. Special topics might include metrics, development activity definitions, and size and cost estimation for object-oriented developments. Status of early adopters and Fast-Track training activities will also be communicated to management and the customer on a regular basis. As enhancements to the Software Development Plan are developed to properly manage object technology, briefings will be held to keep both engineers and managers communicating on the same playing field.

### 3.7.2 Transitioning Products to Object Technology

Object-oriented software must be able to integrated with existing baseline architectures. Some early prototypes, evaluation packages, and subsystem may contain all object-oriented software. Some products will commingle object-oriented software with legacy or procedural software. Where this is the case we will use the technique of reification to provide a well defined interface. Reification is a technique for providing an isolation layer between class structures and procedure or function calls. This technique presents a low risk because it is well understood and used extensively in the industry. The added effort of building reified interface classes is not considered significant. Well defined and controlled interfaces are part of any good software architecture. Reified classes simply encapsulate the software to ensure the interface is enforced at run-time.

We recognize that object models and relational table structures do not match up one for one. In addition, the access mechanisms for the two technologies are fundamentally different. Therefore, where relational database implementations are to be used, mapping rules and guidelines will be developed to ensure data consistency and integrity across the project.

In addition to the software products, guidelines will be developed to produce CDRL products which are "friendly" to the technology and meet all contract requirements. It is know that functional specifications and functional design documents present some mapping difficulties to object-oriented architectures. While this is a relatively low risk area, proper guidelines and annotated outline will ease document production.

### 3.7.3 Transitioning Processes to Object Technology

The OMT methodology is well documented in both the Text "Object Modeling Analysis and Design" by James Rumbaugh et al and the training material provided by the Advanced Concepts Center of Martin Marietta. Since it is a mature methodology used in many fortune 500 companies around the world, by itself, it poses little risk. The challenge is to integrate the methodology with the parts of the program where non-object-oriented techniques and products are employed. Issues to be addressed include program documentation production, control and monitoring metrics and mechanisms, mapping to configuration items, and size and cost estimation techniques.

In the area of metrics, we intend to provide both metrics which are unique to object-oriented software and metrics which can be used in either domain. Some traditional metrics expected values may need to be calibrated in the object-oriented domain. Technical Performance Measures (TPM's) and Program Performance Measures (PPM's) appropriate to object technology will be identified to the risk control board and risk plan.

Reuse is one of the major themes of object technology. We intend to explore the application of the "Experience Factory" concept by Victor Basili. And, we intend to provide process guidelines to engineers on how to:

- apply reification to encapsulate procedural reusable components,
- leverage modeling idioms unique to object technology, and
- use layering with encapsulation to build virtual machine architectures

# 4. Segment Object Technology Plans

In this section, we identify the strategies and risk mitigation plans for introducing object technology into each of the three major segments; Flight Operations, Science Data Processing, and Communications and System Management. This plan is consistent with the program baseline and future technology plans as detailed in the following program documents:

| | |
|---|---|
| 194-813-SI4-002 | Planning and Scheduling Prototype Results Report for the ECS Project, February 1994 |
| FB9401V2 | EOSDIS Core System Science Data Processing Subsystem Reference Architecture White Paper, March 1994 |
| FB9403V1 | Release Plan Content Description White Paper, February 1994 |
| FB9404V2 | Multi-Track Development for the ECS Project White Paper, March 1994 |

This plan will not duplicate other program plans. The intent of this plan is to provide a focal point for identifying potential risks as input to the Risk Management Panel, and, to cost effectively coordinate implementation of object technology. Wherever possible, references will be made to other program plans.

This section of the Transition Plan discusses the current activities taking place on the ECS program to identify and mitigate any risk due to adopting Object-Oriented technology.

The Object-Oriented issues related to ECS success lie in the following areas:

- **Planning.** As with all development efforts, planning is important. However, planning is particularly critical when infusing a new technology.
- **Training.** Assuring that the technical staff is adequately prepared to address the challenges of progressing to the Object-Oriented paradigm is the most important and most easily addressed risk issue.
- **Reuse of Heritage.** ECS must take advantage of every opportunity to increase productivity and reduce risk through the reuse of software that has been developed, tested and proven to be of high quality.
- **New Technology Shock.** Care must be taken to assure that the speed with which ECS adopts Object-Oriented technology does not create unwarranted risk. Experience with the methodology is of prime importance, and techniques such as prototyping and incremental development have proven very beneficial in the past.
- **Measurement and Costing.** The Object-Oriented technology community has had insufficient time to develop "standard" Object-Oriented specific and unique metric values to use for productivity and costing. Care must be taken to assure that planning, scheduling and costing are done sufficiently well to benefit the program. In parallel, it is

necessary to define metrics which can be used to measure the progress of the program and which can be used to provide productivity data for future effort.

Each of these issues is addressed in the context of the overall ECS program in the next section. The following sections address the unique activities being pursued by each of the segments and SI&P.

## 4.1 Issues

### 4.1.1 Planning

Each segment is developing detailed development plans, an integral element of which will be how and where object technology will be introduced. The rationale for the selection or non-selection of the Object-Oriented approach will be detailed in those plans or as adjuncts to those plans. Segment plans will be reviewed by the Object-Oriented Core team to ensure the total risk for the program remains low. Management will be kept informed of the current status of all Early Adopter plans and an overall object technology risk assessment.

### 4.1.2 Training

The ECS training plan provides courses through Martin Marietta's Advanced Concept Center and through the Motorola University. These courses are being taught at the Landover site initially by trained experts from these two companies. The courses are being delivered on a "just in time" basis.

Some courses will "train the trainer", at which point they will shift to being taught by in-house personnel using the same course material. There are several advantages to this approach. First, training can begin with the expertise necessary to carry it out. Then as the source of the instructor transitions, expertise is built up within the contract. This expertise then becomes a valuable resource to be used not only for training, but for application development and for mentoring within segments.

The following OO courses are available to the ECS program personnel, as well as NASA personnel, through the ECS training program:

- System Engineering & Object-Oriented Technology
- Object-Oriented Distributed Systems Design
- Object-Oriented Analysis with StP for OMT
- STP/OMT for Users
- Object-Oriented Database Design
- Object-Oriented Analysis and Design
- Object-Oriented Design
- C++ Programming
- Distributed Processing Using C++

- Advanced C++ Programming

To date about 100 program personnel have taken some combination of these courses. The plan for the program is for all ECS program personnel involved in the Object-Oriented development to be trained in the proper set of courses.

In addition, Loral is hosting C++ training from a local college, and actively sponsoring other training opportunities. One example being a course titled "Overview of Object-Oriented Software Development", which is directed at managers. Furthermore, the Hughes FOS Planning and Scheduling group is hosting some formal and informal training sessions in the Hughes Class Library (HCL).

The FOS group contains several people considered technology and methodology experts in the areas of C++, OOA, OOD and HCL. These key personnel are valuable resources that will be greatly relied on during design and development.

### 4.1.3 Reuse

There are several opportunities to reuse software on the ECS program: reuse of Object-Oriented product software from another program; encapsulation and reuse of non-OO software from another source; reuse of commercially available class libraries; and, internal reuse of ECS developed software.

FOS has the greatest opportunity for reuse among the three segments. A considerable amount of expertise and infrastructure has been developed in this group, including low-level classes and routines packaged together as the Hughes Class Library (HCL). This Object-Oriented library was developed by Hughes for an earlier program. It was designed to be reusable in the general situation of satellite flight operations. The package has been in operation for several years, and has been applied on several projects beyond that for which it was originally designed. The HCL presents the ideal situation for reuse: it is domain specific, it has been designed for reuse, it is Object-Oriented, it has been thoroughly tested, and it has been "wrung-out" in an operational setting. Furthermore, the FOS Planning and Scheduling group is conducting training sessions in HCL.

The Planning and Scheduling heritage systems, on which the FOS Planning and Scheduling system is based, were developed using the OO methodology. A considerable amount of expertise and infrastructure has been developed in this group, including low-level classes and routines packaged together as the Hughes Class Library (HCL). HCL is currently in-house and is being evaluated as an appropriate addition to the ECS project.

The domain expertise of the FOS personnel provides a level of reuse that is not commonly available. Through their experiences with previous development efforts this group will be in a position to draw on/reuse resources which are available throughout their respective organizations.

SDPS will principally be making use of internal reuse via their incremental track development. After the software from one track is subjected to the rigors of testing and user exercises it will be migrated (reused) into the next increment. This internal reuse provides the same advantages as

are inherent in the FOS HCL. The productivity of succeeding increments will improve and the quality of the reused software will increase with use and modification. The SDPS reuse of software within the incremental track development is equivalent to a spiral implementation, and provides the benefits of a spiral methodology.

A principle aspect of reuse that will not be overlooked is that of COTS. The COTS will take the form of standard software packages and reusable Object-Oriented class libraries. The reuse of the class libraries presents an opportunity for productivity and quality enhancement that is not readily available outside the Object-Oriented paradigm.

CSMS is planning to make a great deal of use of COTS products. The character of the segment is such that COTS packages, both standard products and class libraries, are now available. However, the prospects of very powerful tools such as CORBA being available in the near future allow CSMS to postpone their decisions. The planning for these decisions in the context of incremental development allows the opportunity to take the greatest advantage of the increased capabilities.

### 4.1.4 New Technology Shock

A very real risk in the adoption and infusion of a new technology such as Object-Oriented is that so many new issues arise at the same time that the sorting-out of the issues and the application of the proper procedure becomes confusing. This situation is in no way due to the abilities of the organization to deal with the issue or with the suitability of the technology. It is simply a matter of doing the proper engineering to resolve many questions in so short a period of time.

The ECS team is taking five steps to avoid the shock: extensive training, which has already been described; mentoring; and, fast-tracking, prototyping and incremental development.

#### 4.1.4.1 Mentoring

A key element in effectively developing a system using methods or technology in which widely available expertise is lacking is the application of mentors. Mentors are experienced people who are capable of leveraging their expertise and experiences by influencing the direction and efforts of the larger group.

Consultants from the Martin Marietta Advanced Concepts Center have been employed by the program to assist in the infusion of the new Object-Oriented technology into the ECS program. The mentoring is being employed in the following specific areas:

- Development of an Object-Oriented version of the system architecture
- Adaptation of NASA and DoD standard configuration item definitions to the Object-oriented constructs
- Development of DID-compliant Object-Oriented templates for the software documents.
- StP/OMT tool support
- Assistance and guidance in the development and evaluation of Object-Oriented work products

- Development of an ECS metrics program
- Support to the segment managers in all Object-Oriented issues and tasks
- Other Object-Oriented duties as assigned.

FOS has several people considered technology and methodology experts in the areas of C++, OOA, OOD and HCL. These key personnel are valuable resources that will be greatly relied on during design and development.

#### 4.1.4.2 Prototyping, Fast Track and Incremental Development

##### 4.1.4.2.1 FOS

The Off-Line FOS software can potentially derive many of the benefits of the OO methodology. The plan is to develop the off-line prototypes using the OO methodology. This enables a proof-of-concept approach to be applied to the determination of the use of OO for the Off-Line software.

A major concern exists in using OO methodology for the FOS Real-Time requirements. Research and literature on past experiences has shown that there have been some difficulties in meeting real-time needs using OO. Efforts are underway to evaluate the effectiveness of OO for Real-Time activities. Along with the OO training, the Real-Time group has undertaken a prototype effort specifically to evaluate the OO paradigm for a real-time implementation. In addition, the Real-Time group will discuss the use of OO for real-time applications with the ECS OO consultants (i.e., Martin Marietta Advanced Concepts Center), who have experience in the control center domain using the OO methodology.

In addition to Planning and Scheduling, some very small scale prototyping activity is being conducted in the real-time area to explore performance issues associated with implementing OMT models.

##### 4.1.4.2.2 SDPS and CSMS

The incremental development track allows evolution of emerging technology and rapid development of selected ECS software with minimal documentation generated during the development period.

Incremental development is used to mitigate technical risks inherent in software with ill-defined requirements, with extensive interactive software, or with a new technology, or standards heritage. As such, tool kits and some selected ECS components will be developed according to the incremental process. The incremental process will develop the infrastructure of the data management (SDPS) and communication (CSMS) components of the ECS system.

An incremental development approach involves the user community in the process of product evolution, since capabilities are demonstrated frequently in a "build and test a little, evaluate a little" development progression. The direction and progress of the development is verified during each increment --- verifying that user requirements are understood and correctly implemented. Lessons learned in one incremental development cycle may be used to improve

software in the subsequent incremental development cycle. Externally-developed software, COTS, and COTS-intensive software, when available, may be more easily integrated and evaluated than in a more formal environment, due to not being "tied-down" to a specific product early in the development.

The following table summarizes the SDPS prototype plan.

***Table 4. SDPS Prototype Plan (1 of 2)***

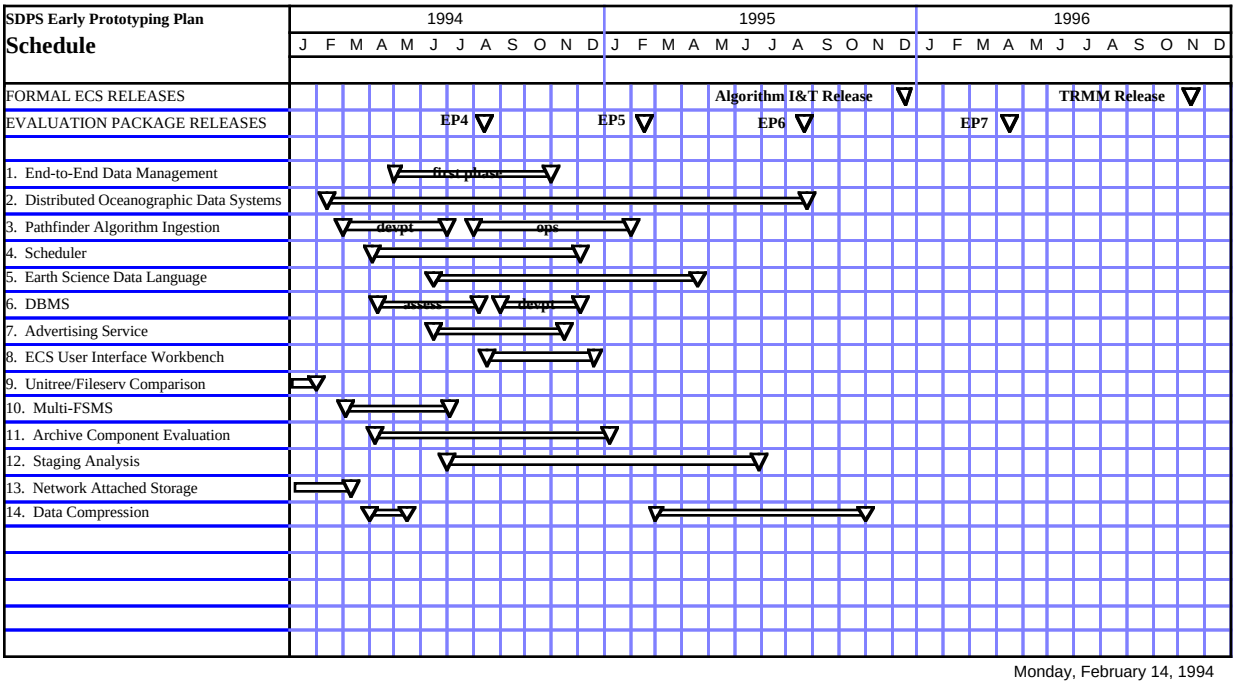
| Prototype | ECS Effort | Other ECS Resources | SDPS External Involvement |
|---|---|---|---|
| 1. End-to-End Distrib-uted Data Management | 12 m/months (for first six months of three year prototype) | STL<br>travel costs | 12 m/months<br>HRL<br>6m/months<br>Sequoia 2000 |
| 2. Distributed Oceanographic Data Systems | 3 m/months | travel to URI | University of Rhode Island |
| 3. Pathfinder<br>Algorithm<br>Ingestion | 6 m/months integration and analysis<br>6 m/months operations | Cray, workstation cluster, SGI + Sony OD reader<br>StorageTek for media storage | GSFC DAAC, HPCC |
| 4. Scheduler | 10 m/months | processing development environment | |
| 5. Earth Science Data Language | 10 m/months | information management development environment | V0 |
| 6. DBMS | 4 m/months initial comparison/vendor benchmarks<br>6m/months ECS investigation of selected candidate | information management development environment | |
| 7. Advertising Service | 7 m/months | information management development environment | CSMS, V0 |
| 8. ECS User Interface Workbench | 8 m/months | SGI or SUN development platform | V0 |

### *Table 4. SDPS Prototype Plan (2 of 2)*

| Prototype | ECS Effort | Other ECS Resources | SDPS External Involvement |
|---|---|---|---|
| 9.Unitree/Fileserv Comparison | complete | complete | complete |
| 10. Multi-FSMS | 12 m/months | archiving development environment | |
| 11. Archive Component Evaluation | 6 m/months | archiving development environment | |
| 12. Staging Analysis | 12 m/months | archiving development environment including RAID disk farm | |
| 13. Network Attached Storage | 6 m/months | archiving development environment | CSMS |
| 14. Data Compression | 8 m/months | archiving development environment | External prototypes (e.g. EDC) |

The following represents the SDPS prototyping schedule.

### *Table 5. SDPS Prototyping Schedule*



### 4.1.5 Measurement and Costing

#### 4.1.5.1 Costing of Object-Oriented Development

The approach to modeling the costing for the ECS system will be to use a combination of the REVIC cost estimating tool and mentoring experience. The REVIC variables which characterize a development effort will be evaluated to estimate the expected differences between an object oriented development and a functional/structured methodology development. The REVIC model is based on Barry Boehm's CoCoMo model and uses line-of-code estimates as its primitive element. Each of the estimates developed for each segment will be subjected to an object oriented-based review to assure that the characterizations of the variables is reasonable, and that the ECS management staff is comfortable with the results.

While it is generally agreed in the object oriented community that line-of-code estimates are not the best measure of the effort required for an object oriented project, it remains that there is no data available which will provide the calibration of the OO costing against organizational experience. It is agreed, however, that the total effort required for an OO project is not inconsistent with that of a functional/structured project. To guard against underestimating the effort required for the OO development costing will be done using conservative line-of-code estimates.

#### 4.1.5.2 Line of Code Count

Line-of-code count has been the Hughes standard metric for costing software development, and was used for costing in the proposal. The OO effort will be developed using C++ code. For the reasons detailed below the actual number of lines of C++ code will be fewer than would be developed in other high-order languages.

The industry experience has been that fewer lines of C++ code are required to provide the same functionality as the same number of lines-of-code in other languages. However, the productivity in developing C++ code for OO applications is lower than other languages and methodologies. Hughes experience in the development of the Planning and Scheduling class library has supported this assertion. This is the same situation that is found in comparing the LOC productivity (LOC/hr.) for assembly language with that of a high-order language (e.g. FORTRAN, C). Far more functionality can be implanted in a system over a given period of time with an HOL than with assembly. However, there will be a far higher LOC count for the assembly than for the HOL. The resultant LOC/hr. will **look** more productive, but more functionality will be available in the HOL application. The reason for this is the relative power of the languages.

#### 4.1.5.3 Object Oriented Metrics

As the development progresses more experience will be gained in the application of the methodology and the C++ language. This experience will be used to refine the REVIC model and the subsequent costing of new elements of the ECS system. The following metrics will be collected during the development:

- Total number of classes

  This will provide a measure of the size of the program at an early stage. The number of classes will be the first element of the design to solidify. The other metrics will evolve as the classes are elaborated as a result of the continuing iterations.

  The total number of cases can be extracted for the StP/OMT tool.

- Total number of methods
- Average number of methods per class

  The Total number of methods and Average number of methods per class can be extracted for the StP/OMT tool.

- Average number of LOC per method

  The Total number of methods, the Average number of methods per class and the Average number of LOC per method will provide a more detailed measure of the complexity of the program, the progress, and the size of the software system, and will provide the first confirmation of the actual number of lines-of-code required.

- Average depth of inheritance tree (DIT)

  This indicates the number of ancestor classes that may affect this class. The deeper the inheritance hierarchy the greater the number of methods it is likely to inherit, and the greater the amount of reuse of inherited methods. There is a trade-off between simplicity (DIT near 1) and maximizing reuse. The rule of thumb is that DIT should be less than 6. The depth of the inheritance also provides a measure of the complexity of the design, and the ease with which the software will be maintained. The more shallow the depth of the inheritance tree the less complex is the design.

  The DIT data can be extracted for the StP/OMT tool.

- Average number of children

  The average number of children is an indication of the influence that a class has on the design. If a class has a large number of children, then it may require more testing of the methods in that class. The greater the number of children the greater the reuse by inheritance, but also the greater the risk of improper abstraction in the parent class.

- Coupling between objects (CBO)

  One class is coupled to another if it uses the methods or attributes of another class. CBO indicates how independent an object is, hence how reusable it is and how easily it will be integrated into the system. The higher the CBO the greater the sensitivity to changes, hence, the more testing is required.

- Average number of attributes per class

  Each of the metrics will be estimated at the beginning of OO design (after SDR) and will be tracked against the emerging actual values as the program progresses. The results will be used as both progress indicators and as emerging actual metric results to be applied later in the program.

  The Average number of attributes per class can be extracted from the StP/OMT tool.

The Software Development Plan reflects an Object Oriented Metrics Plan.

## 4.2 Segment Summary

### 4.2.1 Flight Operations Object Technology Plan

Planning and Scheduling has been using object-oriented programming for their prototype activity. Much of their application will be based on the Hughes Class Library which is written in C++.

Several members of the FOS team attended Object-Oriented Analysis and Object-Oriented Design training .

In addition to Planning and Scheduling, some very small scale prototyping activity is being conducted in the real-time area to explore performance issues associated with implementing OMT models.

Furthermore, the Hughes FOS Planning and Scheduling group is hosting some formal and informal training sessions in HCL.

### 4.2.2 Science Data Processing Object Technology Plan

SDPS is introducing object technology in two area, the Data Acquisition and Distribution Subsystem (DADS) and the Information Management Subsystem (IMS).

The Science Data Processing Segment (SDPS) has a clear need to take advantage of object technology to implement the "Architectural Mandate" presented in the EOSDIS Core System Science Data Processing Reference Architecture white paper, FB9401V2. The philosophy of the reference architecture includes the following key principles:

- open distributed architecture
- behavior or service-orientation
- evolutionary development approach
- support emerging standards and
- forward looking technology integration

The reference architecture will employ an evolutionary migration to object-oriented concepts and object-oriented COTS standards and products. As detailed in the reference architecture white paper, SDPS is employing this evolutionary approach in the following three key areas:

"• the development of objects and services-based system decomposition, as described in the reference architecture white paper, section 3.3.2. In this approach, the objects will be hidden initially under interface classes which represent the various ECS system services.

- interfaces to distributed computing components will use an abstraction layer to allow migration from a client-server based approach (i.e., employing remote procedure calls) to a distributed object approach (e.g., CORBA) with minimal impact on software.
- the database management and search architecture will enable a transition to an object paradigm, again with minimal impact on applications and users."

(taken from the ECS Science Data Processing Reference Architecture white paper)

DADS launched a prototype starting in late 1993 to explore the efficacy of the technology. Since that time their confidence has grown and are in the process of applying OMT to Evaluation package EP-3. This was a low risk approach. Many of the team members were trained in C++ during evening classes. An OMT overview was provided by one of the object-oriented experienced team members. The team was small and all members were experts in the domain. An object model was developed using the Software through Pictures CASE tool (StP/OMT).

### 4.2.3 CSMS

CSMS is pursuing Object-Oriented technology very carefully.

There is much to be gained by judiciously evaluating COTS products and Object-Oriented standards so that the greatest advantage can be taken of the emerging OO tools, standards, and COTS.

CSMS is continuing to pursue the course of incremental development so that the greatest possible benefit can be gained by delaying the implementation of functionality until the COTS products and the reusable class libraries mature, and so that the latest possible technology can be included in the system.

This page intentionally left blank.

# Abbreviations and Acronyms

| | |
|---|---|
| ANSI | American National Standards Institute |
| API | application programmer's interface |
| ASCII | American Standard Code for Information Interchange |
| CORBA | Common Object Request Broking Architecture |
| COTS | commercial off-the-shelf (hardware or software) |
| CSMS | Communications and System Management Segment |
| DBMS | data base management system |
| DCE | Distributed Communications Environment of OSF |
| ECS | EOSDIS core system |
| EOS | Earth Observing System |
| EOSDIS | EOS Data and Information System |
| FOS | Flight Operations Segment |
| GUI | Graphic User Interface |
| H/W | hardware |
| HCL | Hughes Class Library |
| HITC | Hughes Information Technology Corporation |
| HMI | human machine interface |
| IDL | Interface Definition Language |
| IEEE | Institute of Electrical and Electronic Engineers |
| IMS | Information Management System (ECS) |
| ISO | International Standards Organization |
| kbytes | Kilo-bytes ($10^3$) |
| Mbytes | Mega-bytes ($10^6$) |
| NASA | National Aeronautics and Space Administration |
| OO | object oriented |
| OODBMS | object oriented database management system |
| ODL | Object Description Language |
| OSF | Open Systems Foundation |

| | |
|---|---|
| OSI | Open System Interconnect |
| POSIX | Portable Operating System Interface for Computer Environments |
| RDBMS | relational database management system |
| S/W | software |
| SDPS | Science Data Processing Segment |
| SQL | Structured Query Language |
| X.500 | OSI standard for directory services |